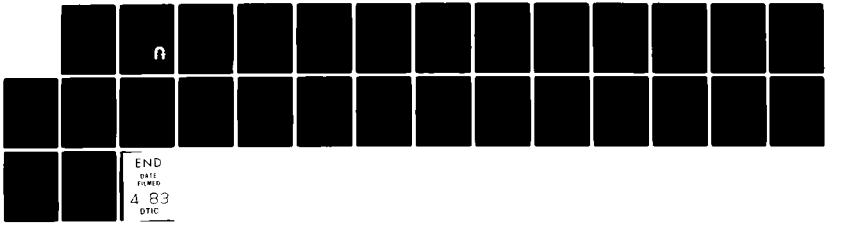
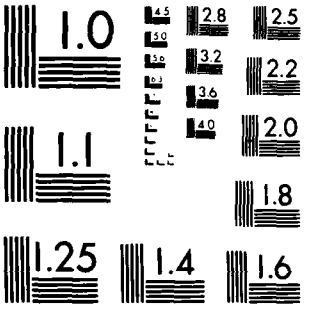
AD-A126 401 DP - COMMAND SET(U) CARNEGIE-MELLON UNIV PITTSBURGH PA 1/1
ROBOTICS INST D GIUSE 10 OCT 82 CMU-RI-TR-82-11

UNCLASSIFIED F/G 9/2 NL



END
DATE FILMED
4 83
DTIC



(12)

Carnegie-Mellon University

DP — COMMAND SET

Dario Giuse

The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

THE ROBOTICS INSTITUTE

CMU-RI-TR-82-11

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: CMU-RI-TR-82-11	2. GOVT ACCESSION NO.: AD-A126401	3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): DP-COMMAND SET	5. TYPE OF REPORT & PERIOD COVERED: Interim
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Dario Giuse	8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Carnegie-Mellon University, The Robotics Institute, Pittsburgh, PA. 15213	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research, Arlington, VA 22217	12. REPORT DATE: 10 October 1982
	13. NUMBER OF PAGES: 23
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for public release; distribution unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DP - Command Set

Dario Giuse

The Robotics Institute
Carnegie-Mellon University
Pittsburgh, PA 15213

10 October 1982

Abstract: This report describes DP, a highly interactive graphics editor that runs on a scientific personal computer. DP is part of a circuit design system that was built at Carnegie-Mellon University to reduce the turnaround time between the conception and the implementation of prototype electronic circuits.

This research was supported by the Robotics Institute, Carnegie-Mellon University.

Table of Contents

List of Figures

Introduction

DP is a highly interactive circuit drawing program that runs on a PERQ computer. It allows a designer to create the description of an electronic circuit in a graphical form that corresponds to the way circuits are normally represented in logic diagrams.

DP is a purely graphic editor: it does not try to "understand" what the user is drawing. All the semantic interpretation is performed by post-processors that are able to extract electrical information out of the drawings. This makes the program itself reasonably simple, giving the designer greater freedom. It also introduces a clear separation between the tool used to create a drawing and the "meaning" of the drawing itself, with the result of a much wider range of possible applications.
Although primarily intended as a tool for drawing electronic circuits, DP may be used as a general illustrating program, able to draw arbitrarily complex pictures.

Chapter 1 explains how to build the program on a new machine and how to start it.

Chapters 2 and 3 contain the essential information; after reading those chapters one should be able to use DP for simple tasks.

Chapter 4 contains details that may be skipped at a first reading.

The Appendix contains a brief summary of the command set.

Version 5.6 is documented in the present manual.

1. DP run-time environment

The following is a description of things one should know before trying to use the program; a short explanation about how to build the program on a Perq and how to start it is also given. A general understanding of the philosophy and operation of the Perq computer is assumed throughout this chapter. The reader should be able to perform the basic operations required to turn a Perq on and to reach the right directory.

Since the interface with the Perq operating system is still rapidly evolving, some of the information in this chapter is likely to become obsolete. Please consult the author for more detailed information.

1.1 Operating system

DP version 5.6 runs under POS, the Three Rivers Computer operating system for the Perq, version D.65. Previous versions of the operating system are no longer supported.

The program may run with both 256 Kbytes- and 1 Mbyte-memory. The smaller memory is not recommended, however, since many of the fast operations must be replaced by slower functions.

1.2 Installing the system on a Perq

Two standard procedures are available for installing the program on a Perq, depending on the physical medium used to transfer the files.

1.2.1 Ethernet

The standard procedure uses the Ethernet transfer program *CMUFTP*. The complete set of files needed to build the system is stored in the CMU-X, the Spice Vax/750. Two different command files allow to retrieve either the whole circuit-design system or DP alone. The two different procedures are listed below[1]:

- retrieve the whole circuit-design system (DP, WLIST, DWL, SL):

```
cmuftp retr /usr/dzg/ALL all
@all
```

- retrieve the graphics editor (DP) alone:

```
cmuftp retr /usr/dzg/DP dp
@dp
```

In order to use one of these procedures the Perq must of course have an Ethernet connection. The previous commands start Cmuftp and retrieve a command file that in turn retrieves all the modules in the system.

---

[1] These are POS commands: type them when you have the POS prompt

1.2.2 Floppy Disk

The alternative distribution medium is a Floppy disk, written in the standard FLOPPY format and available upon request. The disk contains the relocatable files, the Help and Command files for DP, some command files and the definition of the fonts used by the program.

The first file to be retrieved is called getdp.cmd; this command file will in turn retrieve all the others and generate the executable files. Typing

```
floppy get getdp.cmd
@getdp
```

will copy all the files from the floppy to the system disk and will generate the .RUN files.

1.3 Starting the program

Once the program is present on the disk of a Perq it can be started by typing

```
dp
```

This starts DP with default values for all the variables and switches. Such values are currently:

- Current Function = Line;
- Mouse Grid = 3;
- Display Grid = 6;
- Gravity Field = 5;
- Display Scale = 1;
- Current Thickness = 1;
- Current Font = 1 (Gacha7);
- Current Layer = 'STANDARD';
- Color = 1 (Black);
- Pin display OFF.

Typing a filename after "dp" on the command line starts reading commands from an alternate input file; see section 4.6 for further details.

To exit the program type q; answer y to the request of confirmation. Normal exit mechanisms, such as CTRL-C, are disabled by DP to prevent accidental termination.

2. DP: basic concepts

DP is an electronic-circuit drawing tool. It allows the user to interact with a circuit schematic, creating it from scratch or editing an existing one. The output generated by DP may be used as input to post-processors that perform error checking and generate a set of lists, such as a wire list, a stuffing list, a wrap list for wire-wrapping a board, etc..

DP runs on the Three Rivers PERQ computer and is entirely written in Pascal; it uses the high-quality graphic display and the pointing device (mouse) of the Perq. The mouse is the main input device; the keyboard is used for function selection and for typing strings.

DP is a general-purpose graphic editor that can be used to produce drawings other than electronic circuits. It therefore supports advanced graphics operations, such as instance transformations and generic curves, as well as the simpler graphics primitives.

2.1 The basic elements of a drawing

DP drawings are composed of a few primitive elements: lines, circles and arcs, splines, text strings, and pins. All the primitive elements may be combined to form larger elements, called symbols. Figure 2-1 shows examples of primitive elements in DP.

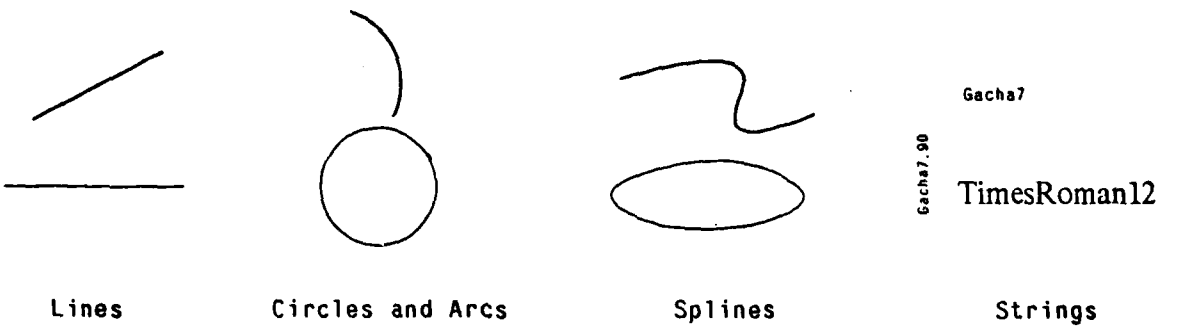


Figure 2-1: DP: primitive graphic elements

The following basic elements are used in DP:

1. **Lines**[2] with have any slope; the endpoints of a line have gravity.

2. **Strings**, composed of ASCII characters; all strings are truncated above a maximum length.

3. **Circles**, and arcs of a circle.

---

[2] finite-length segments, in reality

4. **Splines**, generic curves whose shape is controlled by a set of control points.

5. **Pins**, that is connection points; when used in a symbol pins have gravity.

6. **Symbols**: sets of basic items and possibly other symbols that may be used to represent electronic components. Symbols may be nested, thus allowing hierarchical drawings.

2.2 Status line and Prompt Area

The bottom part of the screen shows the status of the editor. The following items are displayed:

- command character of the function being executed;
- name of the function being executed;
- mouse buttons. Most functions perform different actions for different mouse buttons; the image on the right side of the status line describes the action of every button. The top square stands for the top (or Yellow) button, and so on.

Many commands that require the user to type text use the Prompt Area, located at the lower right-hand corner of the screen. When the program is waiting for keyboard input the area is highlighted; all the prompting commands may be aborted by typing a Ctrl-c (lower-case "c") while the area is highlighted. While the area is highlighted the string may be edited with the standard string edit mechanism; see section 2.5 for more details.

2.3 Mouse buttons

DP usually assigns different meanings to different mouse buttons. The current version of DP uses only three buttons of the mouse; in the case of 4-button mice, the Blue button performs the same function as the Yellow button.

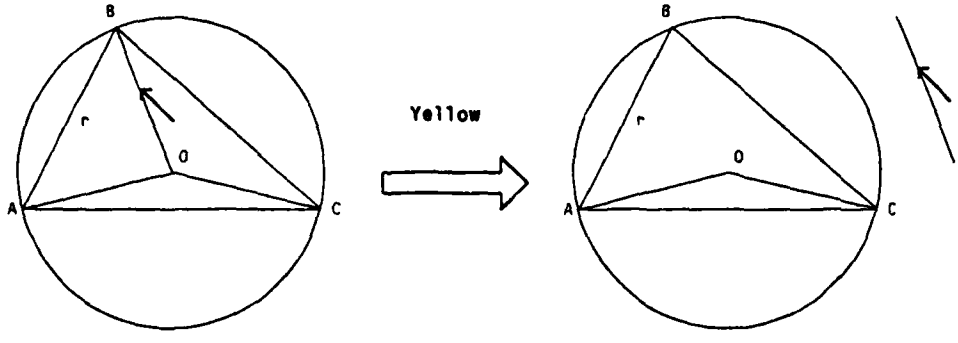


Figure 2-2: Moving one item at a time: **Yellow** button

All the commands that use mouse buttons to form a set of items use the following conventions:

1. The ***yellow*** button creates a set with the single element pointed to by the cursor; the Move

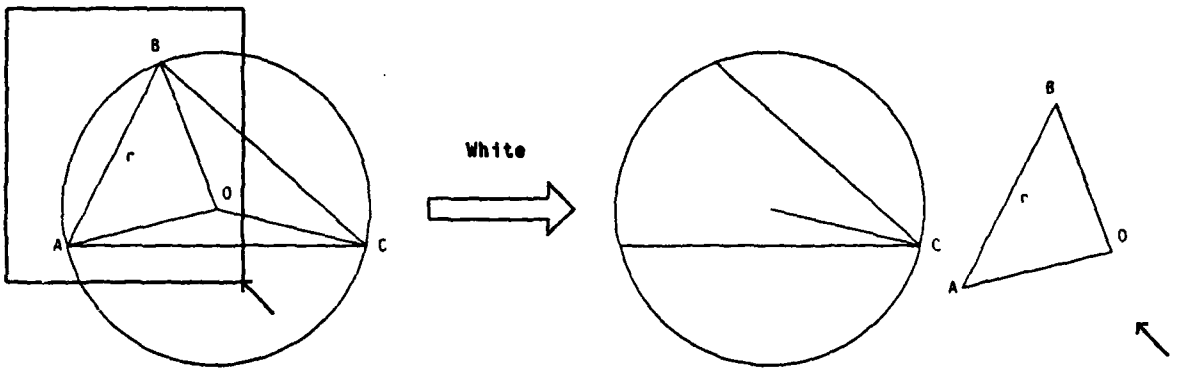


Figure 2-3: Moving the items within a rectangle: **White** button

command, for instance, moves a single item at a time when this button is used. If several candidates exist the smallest object is always chosen; this prevents large objects from "shading" smaller ones.

2. The ***white*** button creates a set with all the items entirely enclosed in a Rectangle[3].
3. The ***green*** button creates a set with all the selected items; for instance using the green button while in Move mode will move all the selected items.

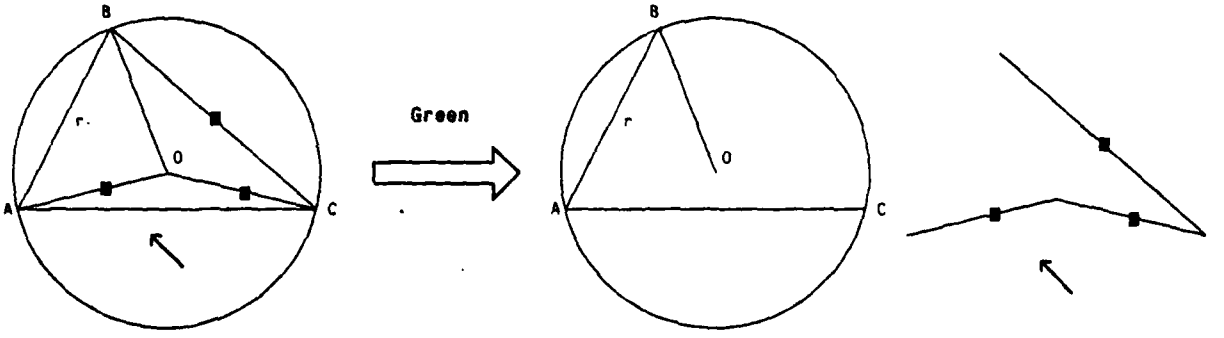


Figure 2-4: Moving the selected items: **Green** button

Figures 2-2, 2-3, 2-4 show the effect of a Move command on the same drawing when the three different buttons are used (the small arrow pointing NW represents the cursor).

Other functions use the mouse buttons differently; this is displayed in the Status line at the bottom of the screen, where each small black square represents a button.

[3] the rectangle is dynamically displayed while the button is held down

2.4 Selecting and deleting

Selection is used to form a set of items that should be handled together. Unlike the temporary sets created by the mouse buttons, the Selected set does not change unless explicitly specified by the user. Selected items are displayed with a small black square near the center of the visible portion of the object. In the case of strings, the whole area occupied by the string is inverted, so that selected strings are displayed as white text on a black background.

To provide some protection against mistakes during **delete** commands, deleted items are not immediately erased; they are kept in a special list that is neither displayed nor affected by normal operations. It is possible to undo the deletion, bringing back the objects.

Because of memory limitations, however, objects cannot be kept around forever. Every time a Delete command is issued all the items that had been PREVIOUSLY deleted are physically erased and their storage is released. In other words, only objects that have been marked as deleted since the last delete command may be undeleted.

2.5 Editing strings

A consistent string-editing mechanism is used throughout DP. This applies both to text strings and to all the commands that prompt the user for a string; in particular it can be used for strings in the Prompt Area. Any time a string is being edited, a special cursor[4] is displayed. Characters are always inserted and deleted at the position following the cursor.

The following characters have a special meaning during string editing:

- BS: delete the character immediately preceding the cursor.
- OOPS: delete all the characters preceding the cursor.
- RETURN: terminate the string editing, i.e. close the string and erase the cursor.

The Mouse may be used to position the cursor within the string and for other functions:

- The **white** button positions the cursor before the character pointed to.
- The **yellow** button positions the cursor before the character pointed to, if possible; the character following the cursor is then deleted. Holding the button down deletes several characters.
- The **green** button terminates the string editing, just as the RETURN key.

2.6 Windows

DP provides a powerful window facility that allows dealing with several drawings without improper interactions between their contents.

---

[4] a thin line between two characters

Each window has a ***separate address space***. The following is a simple way to visualize that feature: a sheet of paper is dedicated to each window, and only the sheet corresponding to the current window[5] may be written on or changed. Operations on a window may affect only the items that have been drawn on that sheet of paper; for example a Select All operation selects all the objects associated with the window. Only some special functions (see section 3.4) may cause an object to "jump" onto another sheet of paper; these functions erase the object from the source window and write it into the destination window.

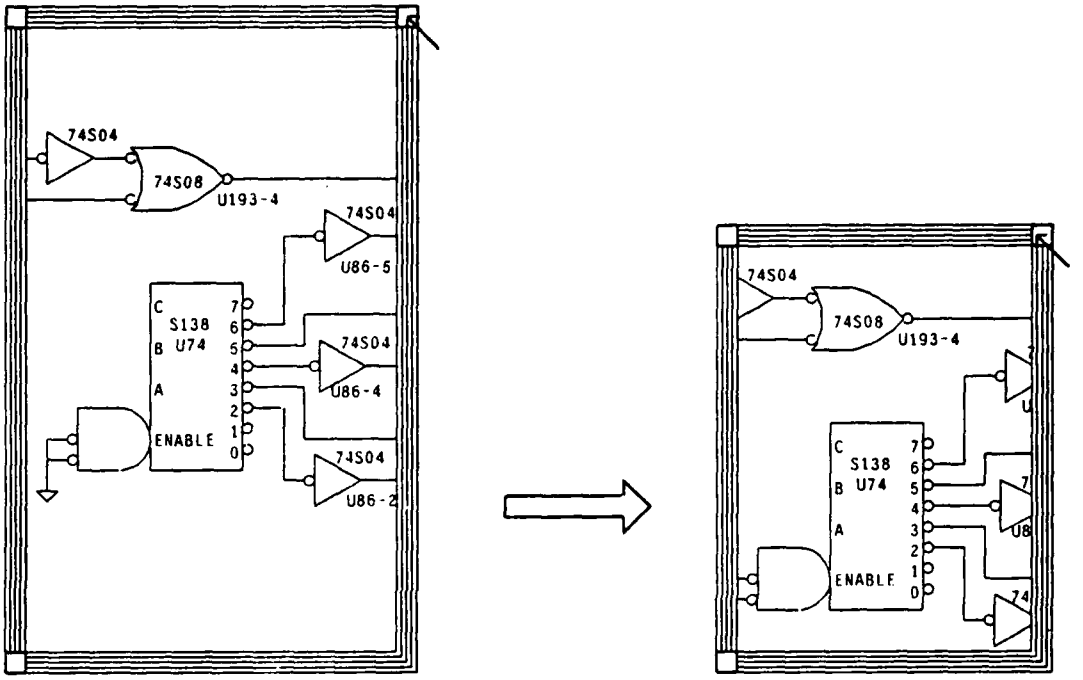


Figure 2-5: Changing the shape of a window

Windows may be manipulated through mouse movements only; the interaction takes place when a button is depressed ***over the border of the window***. Different areas in the border have different functions:

- square at the **upper-right corner**: change the shape of the window. The corner follows the mouse until the button is released.
- square at the **lower-left corner**: create a new window, using one half of the area used by the current window. The new window has a different address space and is initially empty.
- square at the **upper-left corner**: delete the window. The window must be empty; the last window cannot be deleted.
- **gray border**: move the window in the screen, without changing its shape.

---

[5] the window the cursor is in

Figure 2-5 shows how to change the shape of a window by c! ':ing with the mouse over the upper-right corner of the window.

Many commands are disabled when the cursor is not inside a window, since they would not be applicable. It is impossible, for instance, to perform a Select All operation outside a window. If this is the case, the Status Line will not show the new command and a 'beep' will be heard.

2.6.1 Windows and file-names

A region of the border of windows, near the lower left-hand corner, is used to display the file name associated to the window. This is the same mechanism used by Emacs[6]: if no file name is associated with a window when a file is input, the name of the file is displayed and is associated with the window itself. That file name is used as the default name when an output command for the window is issued (both Output and Hardcopy).

If a window has been modified since the last output command, a "*" is appended to the file name. The Quit command checks this flag for all the existing windows and issues a warning if some window is marked as modified. The algorithm used to decide whether a window has been modified is rather conservative (sometimes the window is flagged even if its contents have not actually been changed). The following rules determine the status of the window:

- the Output command clears the Modified flag;
- input commands do not alter the status of the flag;
- all the non-immediate functions set Modified to true;
- the immediate functions do not alter the flag, with the exceptions of Delete and Undelete.

2.7 Layers

DP supports a powerful layering mechanism: a drawing may be thought of as being composed of multiple layers[7], independent of each other. This is intended to provide the same effect as multiple transparencies containing parts of a drawing: single transparencies may be added or removed, modifying the drawing itself.

The layer mechanism is especially useful for complex drawings that contain logically separated parts. An example is a Printed Circuit Board: only one layer at a time is usually being worked upon, but it is essential that all the layers be visible. Setting all the layers but one to read-only constitutes a protection against accidental changes to items on different layers.

The layer mechanism is also used by the circuit post-processors (see [4]) to quickly discard useless information. For instance, the post-processors totally ignore items in the "Comment" layer.

In the case of symbol instances, *layers act as a filter.* If an instance is on an invisible layer, for example, it will

---

[6] A screen-oriented, multiple-windows text editor

[7] The original idea of multiple layers was suggested by Joseph Newcomer

be totally invisible even if it contains items that belong to visible layers. If the instance is visible, on the other hand, the visibility of nested items depends on each item's own layer.

Layers can be manipulated through the Layers menu, accessible through the Unusual Commands menu. The Layers menu displays all the currently defined layers together with the settings of the layer parameters.

The following parameters are associated with each layer, and can be individually set or reset. A black rectangle in the Layers menu means that the parameter is ON.

- **Display**: if ON, the layer is visible. If OFF, the items in the layer are invisible and do not have gravity.
- **Alter**: if ON, the items in the layer are affected by DP operations; if OFF, the layer is write-protected.
- **Output**: if ON, the items in the layer may be written to files, printed and sent to the plotter.

When a layer is set to invisible (Display OFF), it is automatically write-protected.

The Layers menu also displays the Current Layer, i.e. the layer for new items that will be created. The Current Layer may be changed by clicking the Default rectangle of a different layer.

The Layers menu has some buttons at the bottom, with the following functions:

- Create a new layer; the name is prompted for.
- Rename an existing layer (it is illegal to specify an existing name).
- Exit, i.e. resume the normal DP operations.

2.8 Check-pointing

Since typical editing sessions with DP tend to be fairly long, a check-pointing mechanism has been introduced. Each window has a counter associated with it; the counter is incremented every time a new function is entered while the window is active. After a given number of events[8] the whole contents of the window are output and the counter is reset. This ensures that a recent copy of the drawing is always available even in the case of a system crash. The check-point file name is formed by appending '.CKP' to the file-name associated with the window; thus the file "latch.dp" is checkpointed to "latch.dp.CKP".

The number of commands between check-points may be modified by the Unusual Commands menu (see section 3.9).

[8] default 100

2.9 Fonts

DP supports multiple fonts. A "font" consists of the full specification of a font plus the name of a Perq font used to display it. The font specification is device-independent, i.e. it describes an "abstract" font such as Helvetica8 or TimesRoman12.

The description of an abstract font consists of four items:

Face: one or two characters that describe the particular face used. Valid characters are ***{r b i}***: ***r*** means Roman (the standard face), ***b*** means **boldface** and ***i*** means *italics*. Some characters may be combined: for instance, ***bi*** means ***boldface italics***.

Size: an integer indicating the size of the font. Although this number does not have a direct relation to some specific unit, small numbers mean small fonts. See [1] for more details on font lore.

Rotation: an integer indicating the rotation of the font. The rotation angle is measured in minutes, starting from 0 for a standard-oriented font; a font that runs vertically up the page has a rotation of 5400 minutes (90 degrees).

Family: the Ascii name of the family. A font family specifies a set of fonts with similar characteristics: Helvetica, Gacha, TimesRoman etc.

The previous entries are intended to specify the font contents of a drawing unambiguously, and for each output device the best possible approximation will be used.

The Perq font is a particular, device-dependent specification that tells DP what font to use when displaying a string on the Perq screen. It may thus happen that the same Perq font is used to display different fonts in a drawing, when there is no Perq font that matches exactly the different "abstract" fonts. The font information is however carefully preserved in all the drawings and used for different output devices.

3. DP command set

All DP commands are single keyboard letters. No Control key is required, in order to make it easier to type commands with one hand while holding the mouse in the other hand. Upper- and lower-case command letters are considered different. Some of the Perq keys are labeled with a whole word; in the following they are listed in capital letters, e.g. "DEL".

Some commands are ***Immediate***: they are immediately executed when the keyboard command is typed. Such immediate commands do not change the function being executed, and at the end of the operation the previous function will be displayed again in the Status Line of the display.

3.1 Basic items

l — ***Line*** mode. Whenever one of the mouse buttons is depressed a new line is created; when the button is released the line is "frozen". Depending on the button, the line may or may not be constrained to be only horizontal or vertical. If a Gravity buttons is used, both endpoints of the line may be attracted by a gravity point[9]; the first endpoint is attracted when the button is *clicked*, the second endpoint is attracted when the button is ***released***. The Current Thickness is used for the line.

a — ***Ascii String*** mode. Clicking a button causes a string to be prompted for and inserted at the current position. The Green button reads a sequence of strings and aligns them below the first one; the sequence is terminated by an empty line. Every character up to the end-of-line is inserted in the string; strings longer than 80 characters are truncated. The Current Font is used for the string.

0 — ***Circle*** mode. Clicking a button starts a circle with the center in the current position; releasing the button freezes the circle. The Green button creates an arc out of three points: the two endpoints, in counter-clockwise order, and the center. The Current Thickness is used for the circle.

9 — ***Spline*** mode. All the buttons but Green enter a new control point; the Green button creates a curve out of the set of control points. The endpoints of the curve will always lay on the first and last control points; at least two control points are required. The Current Thickness is used for the spline.

p — ***Pin*** mode. Every time a button is clicked a pin is inserted at the cursor position; the pin number is prompted for. If the Green button is used no prompting occurs, and the previous pin number incremented by 1 is used.

e — ***Edit*** items: change the shape of existing items. See 4.3 for the item-specific details of this command; in general, the new shape of the item is always redisplayed dynamically in order to provide an accurate visual feed-back.

---

[9] either the endpoint of a segment or a pin in a symbol.

3.2 Parameters and Fonts

- *(minus)* choose the Current Thickness. The value of this parameter will be used for all the new lines, splines and circles[10].

n use New Parameters for existing items. Alter some or all the parameters of already defined items; for instance, convert strings from one font to another or change the thickness of lines. A menu is used; the previous choice is always suggested as a default. The following parameters may be changed:

- thickness (applies to lines, circles, splines);
- font (applies to strings);
- layer;
- color (applies to all items except symbol instances).

f choose the Current Font or enter a new font in the Font Table. A menu with all the active fonts is displayed; clicking over one of the black rectangles in the Menu selects the new Current Font. The last entry in the menu is labeled "New Font": clicking it will install a new font, reading a Perq font file if necessary, and use it as the Current Font. See section 2.9 for an explanation of the meaning of the font parameters.

3.3 Select and Delete

s enter Select mode. New items are added to the Selected list.

z enter Deselect mode. Items are taken out of the Selected list.

S select all the items in the current window. Immediate function.

Z deselect all the items in the current window. Immediate function.

d physically erase the previously deleted items; enter Delete mode, adding new items to the "deleted" list.

D physically erase the previously deleted items; delete the selected items in the current window. Immediate function.

u undelete the items in the "deleted" list (since the last D or d command), and select them. Immediate function.

---

[10] The valid range for the thickness is 1..8 in the current implementation

3.4 Copying and moving

Move and Copy are the only commands that work across window boundaries, so that an item may be conveniently moved across windows.

m — ***Move***: pick a set of items and move them until the button is raised. A new temporary set is created for each Move operation.

c — ***Copy***: pick a set of items, make a copy, move the copy until the button is raised.

x — ***Stretch***: pick a set of items and move them, stretching all the lines that are connected to the items. The current implementation stretches lines connected to lines, strings or symbols[11] This command tries to preserve all the existing connections and not to create new ones. Strings that represent signal names (on a line) or chip locations (on a symbol instance) are moved with the item they are attached to. *Not completely implemented.*

3.5 Symbol-related commands

These commands manipulate symbol definitions. See also section 4.4.

b — create a set of items and pack them in a new symbol definition. The symbol name is prompted for; entering an empty line generates an automatic name. If the name matches one of the symbols already defined, DP asks whether the old definition should be deleted. In this case references to symbols with the old name are changed to the new definition; this actually means ***redefining a symbol***, and the old definition is destroyed.

B — create a set of symbol instances and unpack them into their basic components; the instances themselves are erased[12]. This function leaves nested symbols and top-level items where the old instances used to be.

t — enter Transformations mode. Symbol instances may be arbitrarily transformed by means of rotate, scale and mirror operations; this command allows to apply a transformation to sets of symbol instances. A Menu is used to specify the desired transformation: a column of black buttons and text is displayed. Clicking over one of the buttons selects the transformation; some of the sub-functions prompt the user for the value of a parameter, e.g. the angle of rotation.

3.6 File I/O

These commands perform various I/O functions. The format of DP drawing files is described in [3]. The commands that deal with drawing files (I i O) automatically append the default extension to the user-typed name.

---

[11] A line is considered to be connected to a symbol if it ends exactly over a pin of the symbol

[12] Unreferenced symbol definitions are garbage-collected.

o — write all the items in the current window to a drawing file; the file name is prompted for, and if a file name is associated with the window it is used as a default. If a file with the same name already exists, it is first renamed with a dollar sign at the end; the file "test.dp" will thus be saved as "test.dp$". This is the standard mechanism used by the Perq editors; it always makes a backup copy of a file before overwriting it.

H — create a hard-copy of the drawing in the current window. A Press file is generated and left on the current directory; the filename is prompted for. The Press file consists of a single page, and may be shipped to the printer later on.

O — send the drawing in the current window to a plotter. The plotter must be connected to the Perq through the RS232 line. Only the HP-7221A plotter is currently supported.

I — read a drawing file and merge it to the items in the current window. To start from scratch, all the items should be cleared before typing this command[13]. The file name is prompted for; if the file does not exist an error message is generated.
If the window is empty this command adds an offset to all the new items, so that the image is centered in the window.

i — read a single symbol definition from a drawing file. Both the symbol name and the file name are prompted for. If the symbol is not defined in the file an error message is displayed; if the search is successful an instance of the symbol is created and centered around the cursor position.

j — read a text file and display it as Ascii strings.[14] The Current Font is used for the new strings.

@ — read commands from an alternate input file whose name is prompted for. The normal operation is resumed when the End Of File is reached; input is directed to the keyboard again.

3.7 Display commands

These commands affect the way the drawing is displayed; they do not change the internal representation of items.

R — redraw the whole screen; this command may be used if the display image has been damaged. Immediate function.

r — redraw the current window. Immediate function.

= — display the current window with a different scale. This command "zooms" the image, it

---

[13] This can be obtained by typing SDD

[14] This command is extremely useful to examine the output of a post-processor, for instance, together with the drawing from which the output was obtained.

does not change the size of items; other windows are unaffected. The scale value is a real number; the default scale is 1. A scale of 0.5, for instance, means that objects are displayed half the real size. Immediate function.

` *(back-quote)* Display Grid on/off; useful for aligning items. Immediate function.

\# display a black square ("diamond") at the connections of lines; two lines that form a corner are not flagged. Immediate command; diamonds are not permanent items.

w move the image in the current window; the image "follows" the cursor and is refreshed at the end. The Green button uses a faster algorithm that requires less startup time.

~ enable or disable the displaying of pin numbers, and redraw the screen.

P enable or disable the displaying of pins, and redraw the screen.

INS insert a Mark at the center of the current window. Each window has a circular buffer of Marks[15] that are used to remember important positions in the drawing. Immediate function.

G go to the next Mark in the circular buffer for the current window; that is, change the position of the window over the items. The next Mark is placed at the center of the window. This command allows one to "jump" between various places in a drawing. Immediate function.

DEL delete the previous Mark. Immediate function.

3.8 Mouse commands

g change the **mouse grid**, that is the distance between the two nearest points the cursor can be in. High grid values make cursor movements gross; small values yield a smooth motion but less accurate positioning. The recommended grid value is 3; this is the best value to use when dealing with normal drawings. The smallest grid value is 1. The value of the Gravity Field is related to the current Mouse Grid: the recommended value is
*gravity = (grid * 2) - 1.*

3.9 Unusual Commands

k This command gives access to a set of rarely used functions; the previous value for the various parameters is always displayed as a default.

- **Checkpoint-value**: number of key-strokes between checkpoints. A checkpoint file is automatically written when more than the specified number of commands is typed in a window.

---

[15] The buffer is initially empty; there is no limit to the number of Marks in a window.

- **Display Grid**: alter the distance between the dots in the Display Grid. This simulates "graph paper" and is useful for aligning items.
- **Gravity Field**: size of the gravity field, in screen units. Smaller fields mean weaker gravity.
- **Clip on window when printing**: indicates whether the whole drawing or just the visible portion should be printed[16] . In the latter case the printed page will look exactly like the screen, without any centering or justification of the image.
- **Diamonds when printing**: indicates whether diamonds should be used at the connections of lines in the hard-copy.
- **Show invisible items**: redisplay the whole screen, showing all the invisible items (including items nested in symbols).
- **Toggle cursor shape**: toggle the direction of the cursor between North-West and North-East.
- **Layers**: invoke the special Layers menu (see section 2.7).

3.10 Miscellaneous commands

?

print out internal information; primarily intended for debugging purposes. This command displays the following items:

- parameters of the current window;
- list of the Deleted items, if any;
- segment allocation table and number of free segments;
- Font Table, with the font numbers and the associated Perq font file;
- names of the Symbol Definitions, if any.

h, HELP

type the help files for DP. There are currently three help files: introductory help, complete command set with a short explanation for each command, and changes in the latest version.

q

quit DP. It asks for a confirmation (type 'y' if you really want to quit, everything else to remain in the program). If some window is marked as Modified and its contents are not null, a new request for confirmation is issued.

[16] applies only to Press files.

4. Advanced topics

This chapter contains some of the advanced techniques a designer needs to know in order to use DP as efficiently as possible.

4.1 The coordinate system

DP uses a standard-oriented, cartesian coordinate system to represent the objects in a drawing. 16-bit integer arithmetic is used throughout, and therefore the integers in the range [-32767..32767] are valid coordinates. No special meaning is assigned to the absolute value of the coordinates in a drawing; translating every item by a fixed amount does not change the drawing at all.

When DP is started the point (0,0) is placed at the center of the window. Both the window and the drawing may be moved, performing any arbitrary translation. As a quick "beam find" operation, specifying a scale of 0 (see 3.7) goes back to the initial position with the window centered around the origin and scale equal to 1.

4.2 The size of a drawing

DP does not restrict drawings to the size of a single screen. Using the window mechanisms one may create drawings that are about 64 x 96 times the size of the screen[17] .

In practice, however, very large drawings are not recommended for output devices like the *Dover printer*. Large drawings must be either split into smaller ones or printed with small magnification. This does not apply to larger-paged output devices, such as plotters.

4.3 Editing items

The action performed by the Edit command depends on the particular kind of item being edited. Items are always opened for editing by pointing at the item with one of the mouse buttons depressed. Circles and Splines are opened by pointing at one of their endpoints; the whole item is "sensitive" in the case of lines, strings, pins, and symbols.

- ***Lines***: clicking over the line picks the nearest endpoint and moves it following the mouse. The line is dynamically adjusted and appears to follow the cursor in a rubber-band fashion. The same options are available as in Line mode: constrained/gravity, etc.

- ***Circles***: clicking over one endpoint with the Yellow button changes the radius; all the other buttons drag the endpoint, changing the subtended angle but keeping the radius constant.
  The circle is always scanned counter-clockwise; the First point marks the beginning of the visible portion of the arc and the Last point marks the end of the visible portion. When the order of the two points is swapped, the circle changes shape abruptly, e.g. from a short arc to an almost full circle.

---

[17] This means, theoretically, a drawing as big as 6000 sheets of paper.

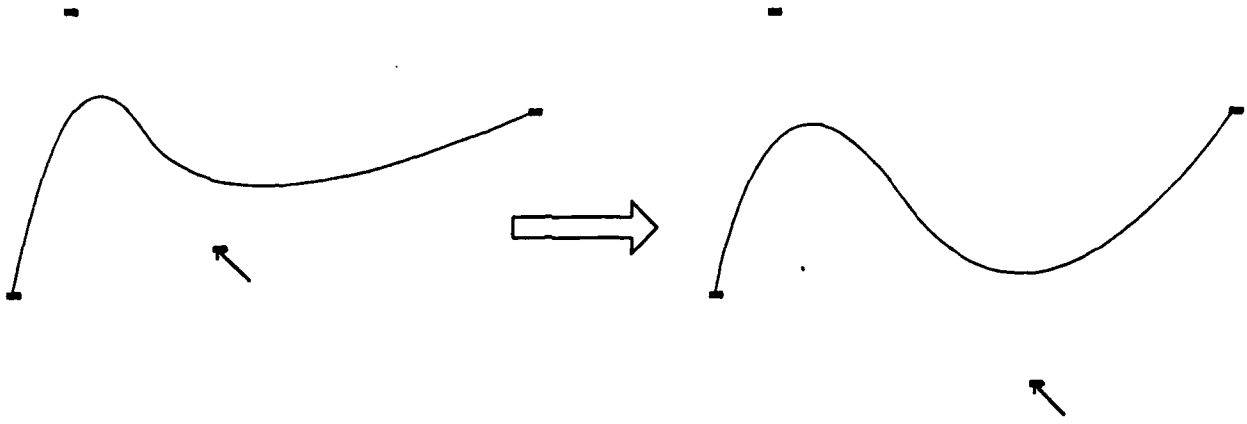


Figure 4-1: Editing a spline by moving a Control Point

If a circle is embedded in a symbol and the symbol is rotated or mirrored, the endpoints of the circle are also transformed. It may thus happen that when the symbol is unpacked the endpoints of a full circle are no longer at (Radius,0); they might be for instance at (0,Radius).

- ***Splines***: after opening the curve for editing (button down over one of the endpoints) the Control Points are displayed. Moving the cursor over one of the control points locks the point to the cursor; the spline is recomputed when the cursor is moved. When the button is released the spline is frozen. Figure 4-1 shows how to edit a spline by moving one of its control points.

- ***Pins***: change the position of the pin number. Clicking over the pin and moving the pointer moves the pin number in one of the four quadrants; Pin Numbers Display should be ON. Notice that the display position for new pins is computed automatically when they are used in a symbol.

- ***Strings***: alter the characters in the string. The string is displayed in the Prompt area and is opened for editing; see section 2.5 for more details. A string cannot be deleted this way: entering a null string simply puts the original string back.

- ***Symbols***: display or alter the name of the symbol. The Green button "opens" the name and allows to modify it; this means that *all the instances* of the symbol will have their name changed, and the old name is deleted. The other buttons simply display the name of the symbol.

4.4 Symbols and Instances

A symbol is a set of items that are grouped together and represent a single object, such as a transistor or a NAND gate; the set is always identified by a unique name.

Although not clearly spelled out, the word "symbol" has been used in this document to mean "a symbol definition". The word "instance" has been used to mean "an instance of an already-defined symbol". To draw an analogy between symbols and programming-language concepts, a symbol (or symbol definition) is equivalent to the definition of a procedure; an instance is analogous to a procedure call.

The symbol defines the structure of a graphical object: a set of basic items and possibly other nested instances. The definition is just a "template", since it specifies how to draw a graphical entity if requested; the definition itself does not add any item to a drawing. It is only by creating instances of that symbol that items will be made visible on the page. See [2] for a discussion of symbols and instances.

An instance, like a procedure call, may specify the value of certain parameters, indicating where and how the graphical symbol is to be displayed. Each instance specifies the following parameters:

- X- and Y-offset: coordinates of the center of the symbol. This specifies the global translation that is to be applied to each item in the symbol.
- Rotation angle: the global rotation of all the items in the symbol. This parameter is in minutes, a positive value meaning a counter-clockwise rotation. A rotation of 30 degrees clock-wise is thus specified as -1800 minutes.
- Scale: two values that specify the global scaling of the symbol and possibly mirroring transformations. A scale of 1 means do not alter the size of items.

4.4.1 Creating a symbol

The best thing to do when creating a new symbol (especially for circuit schematics) is to look closely at existing shapes. A grid value of 3, the standard value, is strongly recommended. A rather high scale, like 3 or 4, should be used when drawing the lines for a shape; when drawing very short lines Gravity should be off.

A good way to create a shape is to edit an old one. To do so, create an extra copy of the old symbol and unpack it; the basic components will be available for editing. When the shape is all right put down the strings[18] and the pins. It is usually better to use the Rectangle button for the final Make Symbol command.

4.4.2 Symbol names

DP allows one to create a symbol without explicitly giving it a name; an internal name[19] is generated. This happens when an empty line is typed as the symbol name.

Symbols with automatic names should not be used at the top level in a circuit drawing, since they cannot be retrieved by name from a drawing. They should be used inside other symbols; as an example consider an integrated circuit symbol. "Pins" in such a symbol are usually complex items: they have a pin, a string (the visible pin name), and possibly other items. Packing these items in a symbol is the basing naming convention for pins used by the circuit post-processors; automatic names are very handy for symbols like this.

---

[18] Do this with scale 1, in order to position the strings correctly

[19] Such as $$12:28:32; this is called an *automatic name*

4.5 Memory allocation

The current operating system of the Perq restricts the total amount of storage that a program may allocate. DP tries to use as much storage as possible for drawings, but sometimes the available resources may be exhausted. Here are some suggestions about memory allocation:

1. Deleted items should always be physically erased when no longer needed. Simply deleting an item does not release the associated storage, since the item may still be "undeleted"; it is thus necessary to use Delete commands twice, since this physically destroys the deleted items. To get rid of a whole drawing, for instance, one should type "SDD": Select All, Delete All, Delete All (the second D releases the storage).

2. Working with many complex drawings at a time requires large amounts of storage; unneeded drawings should be promptly deleted.

3. Using many fonts requires several data segments to be permanently allocated; once a font has been installed it is not released until DP exits.

4.6 Alternate input for commands

There is a mechanism for reading commands from a transcript file instead of from the keyboard. A command file is read via the @ command and should contain the same commands that would be typed on the keyboard, plus cursor-positioning commands.[20] The alternate input file mechanism may conceivably be used as a "macro" facility to perform simple initializations of switches and parameters.

---

[20] Transcripts of DP sessions may be created with the $ command; this command alternatively opens and closes a transcript file, called "dpScript" The $ command is not yet officially supported.

I. Command set table

a enter Ascii String mode.
b create a symbol definition.
B unpack symbol instances into their components.
c copy items, move until the button is released.
d delete items.
D delete all the selected items.
e enter Edit mode.
f choose the Current Font or enter a new font.
g change the mouse grid.
G go to the next Mark in the circular buffer.
h (or HELP): type the Help file.
H create a hard-copy of the current window.
i read one symbol definition from a file, put it at the current position.
I read a drawing from an input file, merging it with the current window.
j read a text file, create strings.
k unusual commands.
l enter Line drawing mode.
m move items.
n force new parameters for existing items.
o write the contents of the current window to an output file.
p enter pin mode.
P toggle displaying of Pin Positions.
q quit DP.
r redraw the current window.
R redraw the whole screen.
s enter select mode.
S select all the items in the current window.
t Transform symbol instances.
u Undelete items deleted since the last delete command.
w move the image inside the current window.
x pick and move items, stretching connected lines.
z enter deselect mode.
Z deselect all the items in the current window.
0 enter Circle drawing mode.
9 enter Spline drawing mode.
~ toggle the displaying of pins.
` toggle the Display Grid.
= enter a new scale for the current window.
display Diamonds at the intersections of lines.
- enter the Current Thickness.
? print out internal information.
INS insert a Mark at the center of the current window.
DEL delete the last Mark.

References

[1] Robert F. Sproull.
Font Representations and Formats.
Technical Report, XEROX - Palo Alto Research Center, October, 1980.

[2] William M. Newman, Robert F. Sproull.
Computer Science Series: Principles of Interactive Computer Graphics.
McGraw-Hill, 1979.

[3] Dario Giuse.
DP - Format of the drawing files.
Technical Report, Carnegie-Mellon University, 1981.

[4] Dario Giuse.
SL: a hierarchical wire-lister for DP drawings.
Technical Report, Carnegie-Mellon University, March, 1982.